On the Verification of Wireless Transaction Protocol Using SGM and RED

Pao-Ann Hsiung, Farn Wang, and Ruey-Cheng Chen
Institute of Information Science, Academia Sinica, Taipei 115, Taiwan, ROC
E-mails: hpa@computer.org, farn@iis.sinica.edu.tw

Abstract

The exponentially large sizes of state-spaces have been a major obstacle in formal verification, which are due to high process concurrency, and large constants that are compared to clock variables and to discrete variables. We show how an intelligent permutation of reduction techniques and a good selection of data-structures can be used to decrease the effect of the above explosion factors. First, a well-accepted experiment platform for the scalable verification of real-time systems, called State-Graph Manipulators, *is used to verify* Wireless Transaction Protocol *(WTP), which is a part of a fast-permeating world standard,* Wireless Application Protocol *(WAP). Application results show how SGM handles large clock constants and large discrete constants efficiently. Second, a recently proposed* Region Encoding Diagram *(RED) technology is used to show how state-space size explosions due to high concurrency can be efficiently handled in WTP verification.*

1. Introduction

Formal verification is often hindered by exponentially large sizes of state-spaces, which can be accounted to three factors: a high degree of system concurrency, large constants that are compared to clock variables, and large constants that are compared to discrete variables. To overcome the problem of size explosions of state-spaces, several reduction techniques and data-structures have been proposed in the literatures [3, 5, 7, 12, 13, 14, 15, 16] and implemented in various verification tools [2, 4, 6, 8, 9, 12, 16, 13]. A state-space reduction technique tries to decrease the effect of only one of the above three explosion factors. In general, a combination of reduction techniques must be applied to successfully verify a real-time system. We will show how the *compositional* model-checking framework in *State-Graph Manipulators* (SGM) [12, 13, 16] tool allows experimentations with different combinations of reduction techniques. Our target application is a transaction layer protocol, called *Wireless Transaction Protocol* (WTP), which is a part of the *Wireless Application Protocol* (WAP) [10]. We will also show how the application of *Region Encoding Diagrams* (RED) [8, 9] to WTP verification can efficiently reduce state-space explosions caused by concurrency.

SGM is a high-level real-time system verification tool. It provides a flexible experiment platform for applying different sequences of reduction techniques to a real-time system state-space. In SGM, the verification framework is *model-checking* and the state-space generation is *compositional*. Each reduction and composition technique is modularized into a re-usable *manipulator* that acts on *state-graphs*, a high-level representation of state-space. SGM is further described in Section 3.

RED is a recently proposed BDD-like data-structure for the fully symbolic verification of real-time systems. Unlike in DBM (*Difference-Bound Matrix*) [1] which records differences between pairs of clock readings, RED records the ordering among fractional parts of clock readings into encoded integer sequences. Like in DBM, RED has a minimal canonical form and is efficient. Several experiments carried out using the RED tool [9], based on the RED data-structure, show how RED can efficiently handle state-space explosions due to concurrency. RED is further described in Section 4.

Using the experiment platform provided by SGM, we show how the state-space size of a system executing WTP is affected by two discrete parameters, namely, a maximum retransmission counter and a maximum re-acknowledgment counter. Further, we also show that different sequences of manipulators reduce a space-space to different extents and hence we can select an optimal sequence for WTP. SGM can handle large constants efficiently, but not high concurrency. We also show how concurrency in WTP can be efficiently handled by the use of RED technology.

This paper is organized as follows. Section 2 describes *Wireless Application Protocol*, including details on the specification of *Wireless Transaction Protocol*. Section 3 introduces *State-Graph Manipulators*. Section 4 introduces

1530-1427/00 $10.00 © 2000 IEEE

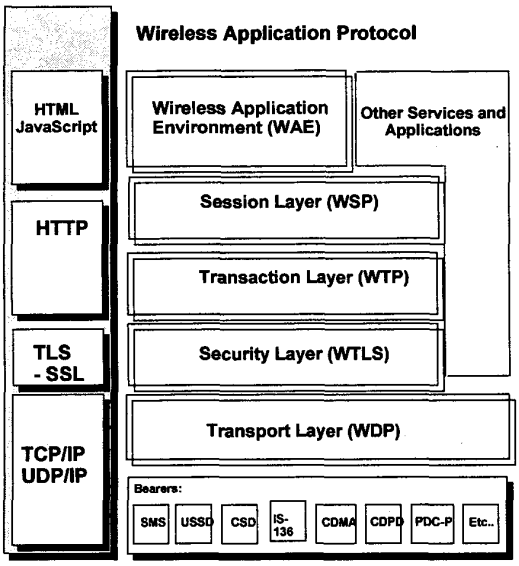

Figure 1. WAP Protocol Stack [10]

Region-Encoding Diagrams. Section 5 gives results on the verification of WTP and shows how SGM and RED allow highly scalable verification of WTP, in terms of efficient handling of large constants and high concurrency, respectively. Section 6 concludes the article.

2. Wireless Transaction Protocol

The WAP protocol stack consists of six layers: application, session, transaction, security, transport, and network. *Wireless Application Environment* (WAE), *Wireless Session Protocol* (WSP), *Wireless Transaction Protocol* (WTP), *Wireless Transport Layer Security* (WTLS), *Datagrams* (UDP/IP), and *Wireless Bearers* such as GSM, CDMA, etc. correspond to the protocols, functions, or media used in the six layers, respectively, as shown in Fig. 1.

We will concentrate on WTP as our verification target. WTP is a protocol defined to provide the services necessary for interactive "browsing" (request/response) applications. WTP runs on top of a datagram service and optionally a security service. Three classes of transaction services are defined: Class 0: Unreliable invoke message with no result message, Class 1: Reliable invoke message with no result message, and Class 2: Reliable invoke message with exactly one reliable result message.

For illustration purposes, we will concentrate on Class 1 transactions. In WTP terminology, a WTP provider that invokes a message is called an *Initiator* and a WTP provider that receives a message is called a *Responder*. A WTP provider may sometimes act as an initiator and sometimes as a responder. Besides initiator and responder, we add a network entity that synchronizes the two WTP providers. Thus, three timed automata (see Section 3) are specified for modeling a WTP transaction instance.

Basically, one invoke message is sent from the Initiator to the Responder. The invoke message is acknowledged by the Responder. The Responder maintains state information for some time after the acknowledgement has been sent to handle possible re-transmissions of the acknowledgment if it gets lost and/or the Initiator re-transmits the invoke message. At the Initiator, the transaction ends when the acknowledgement has been recieved. The transaction can be aborted at any time.

Each transaction has an identifier assigned as soon as it is invoked by the Initiator. At the Responder, when a received transaction identifier is not a valid one as compared to that in the buffer, then verification of transaction identifier is requested by the Responder. This is done by sending an acknowledgment message to the Initiator. If the transaction identifier is valid, then the Initiator sends an acknowledgment back to the Responder, otherwise the Initiator aborts the message.

3. State-Graph Manipulators

State-Graph Manipulators (SGM URL: `http://www.iis.sinica.edu.tw/~eric/sgm/`) was created because most existing tools and methodologies do not provide flexibility in manipulating state-spaces and in applying reduction techniques to a specific real-time system verification instance. SGM has been applied to various benchmarks such as Fischer's timed mutual explosion protocol, CSMA/CD protocol, token ring protocol, etc. The compositional approach adopted in SGM allows scalable verification of real-time systems.

SGM adopts and presents a high-level view of all verification intricacies. Various different sophisticated verification technologies are packed into efficient *manipulators* that act on high-level data-objects representing state-spaces. These data-objects are called *state-graphs*. With a friendly graphical user interface, users can choose manipulators during state-graph constructions and compare the effects of applying different *sequences of manipulators* to state-graphs. As a result, users have full flexibility in controlling the sizes of state-graphs and thus verification scalability.

In the rest of this section, we will give some informal definitions and introduce the reduction techniques implemented as manipulators in SGM.

A timed automaton (TA) is composed of various *modes* interconnected by *transitions*. Variables are distinguished into *clock* and *discrete*, where variables of the former type

increment at a uniform rate and can be reset on a transition, while variables of the latter type change values only when assigned a new value on a transition. A TA may remain in a particular mode as long as the values of all its variables satisfy a *mode predicate*, which is a conjunction of clock constraints and boolean propositions.

As far as temporal property specification is concerned, SGM uses TCTL. A TCTL formula has the following syntax $\phi ::= \eta | \exists\Box\phi' | \exists\phi' \mathcal{U}_{\sim c}\phi'' | \neg\phi' | \phi' \vee \phi''$. Here, η is a mode predicate in $B(C, D)$, ϕ', ϕ'' are TCTL formulae, $\sim \in \{<, \leq, =, \geq, >\}$, and $c \in \mathcal{N}$. Due to page-limit, we do not elaborate on the semantics of a TCTL formula. Details can be found in [11].

A *region* is a collection of states such that: (1) they have the same *mode-vector label* (μ), which gives the mode names that each process timed automata is in, (2) they satisfy the same timing constraint in the form of a *zone* (ζ), which is a *Difference-Bound Matrix* (DBM) [1], and (3) they have the same *set of literals* (L), representing the discrete variables' valuation. Thus, a region can be uniquely represented by a triple (μ, ζ, L).

A manipulator checks the characteristics of each region in a state-graph, performs some comparisons between region characteristics or between regions themselves, merges identical regions/transitions, removes redundant regions/transitions, etc. In the following, we mention five manipulators, of which only the timed symmetry reduction is not applicable in WTP verification because the three automata are independent and asymmetric.

- **State-Graph Merging**: The **merge_graph()** manipulator is a construction technique for composing two state-graphs into a single state-graph that represents the concurrent behaviors of the two components.
- **Timed Symmetry Reduction**: This is an extension of the *untimed* symmetry-based reduction technique proposed by Emerson and Sistla [7], such that *clocks* and their *readings* are also considered for symmetry reductions. We have implemented our *timed-symmetry reduction technique* using a normalization scheme in the form of a manipulator called **normalize_region()**.
- **Clock Shielding**: The **shield_clock()** manipulator works as follows: if a clock is not in a TCTL specification and if it is never read before it is reset, or will never ever be read again, then the clock can be shielded.
- **Read-Write Analysis**: By analyzing all the possible values that a process automaton writes to, or does not write to a global discrete variable, this manipulator, called **read_write()**, computes the literal set of each region. The literal set represents the values that each global discrete variable does not possibly have in a region. Through such an analysis and literal sets formulation, transitions that have triggering conditions conflicting with the literal sets will never be triggerable and can thus be eliminated from further consideration.
- **Bypass Internal Transition**: BIT is a reduction manipulator that detects and bypasses a transition in an intermediate state-graph, that is not observable by external processes.

4 Region Encoding Diagrams

Region Encoding Diagram (RED URL: `http://www.iis.sinica.edu.tw/~farn/red/`) was first proposed for symmetric real-time systems with a single clock per process [8]. Recently, it was further extended for asymmetric real-time systems with unrestricted number of global and local clocks [9]. Compared to the classic DBM [1], RED provides data-sharing capability of fully symbolic manipulation. The number of variables used in RED is $O(|X|log|X|)$ when X is the clock set in the input system description. Experiments show that RED compares more favarobly than Uppaal2k and Kronos, as far as scalability in concurrency handling is concerned.

Considering some benchmarks with which RED has been experimented, it can handle 20 processes obeying the Fischer's timed mutual exclusion protocol, 17 processes running the CSMA/CD network communcation protocol, and 29 processes executing the FDDI token-passing ring-network protocol. All the benchmark results show RED as a promising technology for implementing future model-checkers. Currently, most model-checkers are based on the DBM technology, which requires two types of data-structures: BDD and zone matrices to handle discrete and clock variables, respectively.

Currently, RED is still in its infancy, and hence when large constants are compared with clock variables, it might not compare well with other verification data-structures, such as DBM. In such a case, we use the DBM-based SGM for verifying WTP. Details of RED are out of scope here, interested readers are advised to refer to [8, 9].

5 Experiments on Verification Scalability

In this section, we will show how different tools (reduction techniques and data-structures) increase verification scalability in different ways. SGM can efficiently handle large clock and discrete constants, while RED can efficiently handle high degree of concurrency. Experimenting with WTP, the above two tools were applied and results obtained to support our claim.

We verified the WTP Class 1 transactions for all the three different bearer configurations given in the WTP Specification, namely, GSM (Global System for Mobile communica-

Table 1. Bearer Type Configurations

Bearer Type	R	A	W	P_{max}	Q_{max}
GSM SMS	40	5	300	4	4
GSM USSD	15	5	60	4	4
IP	4	1	40	8	6

tion) SMS (Short Message Service), GSM USSD (Unstructured Supplementary Service Data), and bearers supporting Internet Protocol (Circuit switched, Cellular Digital Packet Data, ...). The parameters used for modeling these three types of bearers are given in Table 5, where R is the retry time-out interval, A is the acknowledgment time-out interval, W is the wait time-out interval, P_{max} is the maximum bound on invoke retransmissions, and Q_{max} is the maximum bound on re-acknowledgments.

All experiments were carried out on a Linux box with a Celeron Pentium III/300 MHz CPU and 256 MB of physical memory. The tool versions used were: SGM version 1.4 and RED version 2.0.

5.1. Efficient handling of large constants

First, we used SGM for verifying WTP. Totally, four manipulators were applied to the WTP specification, namely, **merge_graph** (mg), **read_write** (rw), **shield_clock** (sc), and **bypass_internal_transition** (bit), which were introduced in Section 3. The size of the state-spaces, along with execution time and memory requirements, are given in Table 2 for GSM SMS. Due to page-limits, state-space sizes for GSM USSD and IP are omitted.

Table 2. State-Space Reductions for WTP on GSM SMS

Manipulator Sequence	State-Space Size		Memory (MB)	Time (sec)
	# M	# T		
mg	558	851	0.87	0.25
mg, rw	543	836	1.20	1.06
mg, sc	336	582	0.75	0.22
mg, rw, sc	332	570	1.02	0.97
mg, sc, rw	328	560	0.95	0.72
mg, rw, bit	347	621	1.26	1.62
mg, sc, rw, bit	**199**	**450**	1.03	1.09
mg, rw, sc, bit	201	451	1.08	1.32
mg, rw, bit, sc	207	461	1.08	1.29

 # M = Number of modes, **# T** = Number of transitions

From the application results, we observed that the sequence $\langle mg, sc, rw, bit\rangle$ achieves the smallest global state-graph in all the three bearer configurations. Hence, we can basically conclude that it is the best sequence in terms of state-space reduction for WTP, irrespective of the bearer type. Looking at the parameters in Table 5, we can observe that GSM SMS and GSM USSD both have the same two counter values, P_{max} and Q_{max}. The values of the two parameter counters were larger in the IP case, and so also were the state-space sizes. Thus, we can say that the size of WTP state-space varies with the value of the two counters, instead of the clock intervals. This observation is also further supported by the result that a large difference in the wait time-out intervals ($W = 40$ for IP and $W = 300$ for GSM SMS) did not affect the state-space sizes.

Table 3. Efficient Clock Constants Handling by SGM

R	A	W	SGM (v1.4)		RED (v2.0)	
			M	T	M	T
1	1	1	0.77	0.18	0.33	9.60
5	1	1	0.90	0.24	0.88	28.32
1	5	1	2.07	0.81	0.87	60.58
1	1	5	0.78	0.20	0.99	35.57
5	5	1	0.81	0.20	0.52	41.55
5	1	5	0.85	0.21	0.90	28.07
1	5	5	1.99	0.82	0.88	60.67
5	5	5	0.77	0.18	0.51	40.24
14	5	60	0.91	0.26	4.38	1375.00
40	5	300	0.87	0.24	Not Available	

M = Memory (MB), **T** = Time (s), $P_{max} = Q_{max} = 4$

We further compare SGM and RED in their capability to handle clock constants, by varying the values of the three clock intervals in WTP. From the results shown in Table 3, we observe that SGM handles large clock constants much better than RED. The RED results for last row is not shown because it took more than an acceptable amount of time for state-space construction. The last two rows are respectively parameter values for GSM USSD and GSM SMS. It can be observed that SGM completes state-space construction for both the bearer types using little CPU time and memory space, while RED fails for GSM SMS. This is due to the large clock constant (300) for wait time-out interval (W).

5.2. Efficient handling of high concurreny

Since SGM is based on the DBM technology, small differences in clock intervals did not affect the state-space size or verification time and memory requirements, but differences in concurrency does have an effect. Hence, we also experimented with a new BDD-like data-structure, called *Region Encoding Diagram* (RED) [8, 9], by applying it to the WTP specification. We varied the two counter values to

Table 4. Efficient Concurrency Handling by RED

P_{max}	Q_{max}	SGM (v1.4)		RED (v2.0)	
		M	T	M	T
4	4	0.77	0.20	0.33	9.58
6	6	1.05	0.30	0.34	13.87
10	10	1.63	0.70	0.36	23.20
20	20	3.59	3.38	0.37	46.67
40	20	4.57	5.50	0.37	46.67
40	40	10.51	28.48	0.38	98.42
60	60	22.64	108.56	0.40	156.52
80	80	40.27	296.71	0.43	221.74
100	100	65.65	726.90	0.46	293.91

M = Memory (MB), T = Time (s), $R = A = W = 1$

model different degrees of concurrency. All time-out intervals were fixed at a very small value of one, so that we could observe the effects of concurrency alone. From the results shown in Table 4, we observe that a higher concurrency (as implied by larger counter values) did not significantly increase memory requirements for RED (0.33 MB to 0.43 MB), while the same is not true in the SGM case (0.77 MB to 65.65 MB). As for the execution time, SGM performed very well when the concurrency was low, but it took much longer than RED when the concurrency was high. All these observations go to show that with a good selection of data-structure such as RED, verification can be scaled to larger, highly-concurrent systems.

6 Conclusion

We have shown how formal verification can be made more scalable through a clever choice of reduction techniques (manipulators in SGM) and of data-structures (RED). Different reduction techniques focus on different factors that cause state-space explosions such as concurrency, clock constants, and discrete constants. SGM, based on the conventional DBM data-structure, handles large constants efficiently, while RED, based on the new BDD-like diagram data-structures, handles concurrency efficiently. We have shown how SGM and RED can be useful in the detection and choice of reduction techniques for the *Wireless Transaction Protocol*, a part of the world standard, *Wireless Application Protocol* (WAP).

References

[1] R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *Proc. IEEE Intl. Conf. Real-Time Systems Symposium (RTSS'92)*, 1992.

[2] J. Bengtsson, F. Larsen, K.and Larsson, P. Petterson, Y. Wang, and C. Weise. New generation of UPPAAL. In *Procs. of the Intl Workshop on Software Tools for Technology Transfer (STTT'98)*, July 1998.

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5th Annual Symposium on Logic in Computer Science*, June 1990.

[4] A. Cimatti, F. Clarke, E.and Giunchiglia, and M. Roveri. NuSmv: a reimplementation of smv. In *Procs. of the Intl Workshop on Software Tools for Technology Transfer (STTT'98)*, July 1998.

[5] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *Lecture Notes in Computer Science*, volume 697, 1993.

[6] C. Daws, A. Olivers, S. Tripakis, and S. Yovine. The tools KRONOS. In *Hybrid System III, Lecture Notes in Computer Science*, volume 1066, pages 208–219, 1996.

[7] E. Emerson and A. Sistla. Utilizing symmetry when model-checking under fairness assumptions: An automata-theoretic approach. *ACM Trans. on Programming Languages and Systems*, 19(4):617–638, July 1997.

[8] W. Farn. Efficient data structure for fully symbolic verification of real-time software systems. In *Procs. International Conference on Tools and Algorithms for the Construction of Applications and Systems (TACAS'2000), Lecture Notes in Computer Science (LNCS)*, volume 1785. Springer-Verlag, March 2000.

[9] W. Farn. Region encoding diagram for fully symbolic verification of real-time systems. In *Procs. IEEE Computer Software and Applications Conference (COMPSAC'2000)*. IEEE CS Press, October 2000.

[10] W. A. P. Forum. Wireless application protocol specifications. Version 1.2 available on the web site at http://www.wapforum.org/, 1999.

[11] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. In *Proc. IEEE Logics in Computer Science*, 1992.

[12] P.-A. Hsiung and F. Wang. A state-graph manipulator tool for real-time system specification and verification. In *Procs. of the 5th Intl Conf on Real-Time Computing Systems and Applications (RTCSA'98)*, October 1998.

[13] P.-A. Hsiung and F. Wang. User-friendly verification. In *IFIP TC6/WG6.1 Joint International Conference on Formal Description Techniques For Distributed Systems and Communication Protocols & Protocol Specification, Testing, And Verification, (FORTE/PSTV '99)*, October 1999.

[14] D. Peled. All from one, one for all: On model checking using representatives. In *Proc. of the 5th Intl Conf on Computer-Aided Verification, Lecture Notes in Computer Science*, volume 697, pages 409–423, 1993.

[15] S. Tripakis and S. Yovine. Analysis of timed systems based on time-abstracting bisimulations. In *CAV'96, Lecture Notes in Computer Science*, volume 1102, 1996.

[16] F. Wang and P.-A. Hsiung. Automatic verification on the large. In *Proc. 3rd IEEE High-Assurance Systems Engineering Symposium (HASE'98) – (invited paper)*, November 1998.